

Kiswahili Linux Localization.

Bringing Kiswahili into the world of the Information and Communication Technologies.

22nd April 2004

Dr. H.M. Twaakyondo
<htwaaky@cs.udsm.ac.tz>

University of Dar es Salaam
Dar es Salaam. Tanzania

Dr. A. Escudero-Pascual
<aep@cs.udsm.ac.tz>

Royal Institute of Technology
Stockholm. Sweden

Abstract

Any nation using its own language as the national one has to consider herself as a nation of self confidence and identity. Tanzania with a population of more than 30 million people is proud of using its natural language Kiswahili in the daily communications. However, Kiswahili is not present in the potential and promising sector of the new Information and Communication Technologies (ICT). The paper describes the relevance of Kiswahili as a regional language and the need of bringing Kiswahili into the Information and Communication Technologies sector. The paper presents an initiative to localize Open Source Software including Linux to Kiswahili and a proposed methodology to accomplish this challenge.

Keywords: Kiswahili, ICT, Open Source Software, Localization

Introduction

The Kiswahili language is widely spoken, and it has its historical development. In [1] explores the origin and how Kiswahili was adopted as a national language. From its adoption as Tanzania's national language in 1962, Kiswahili has been used in the primary education system while English is used in secondary schools to university. As a result, Kiswahili is just taught as a subject from the secondary school.

In recent years, the Ministry of Education is considering to change this situation and extend the Kiswahili education to secondary and high schools. The introduction of Kiswahili as a learning language in all levels of education also requires that the tools and educational content are available in Kiswahili.

The Department of Computer Science in collaboration with the Kiswahili Research Institute at the University of Dar es Salaam (UDSM) has initiated a process to create the necessary human network of researchers and interested institutions to carry out the localization of the main components and applications of the Linux Operative System to Kiswahili. To fulfill this goal both research and implementation components needs to be properly addressed.

A project of this nature and size not only requires a proper methodology but a set of auxiliary resources as English-Kiswahili technical glossaries in place.

KiLinux: A localized Operative System for Kiswahili speakers

Tanzania's National Development Vision 2025 [2] was formulated through a process of seeking views and thereby building a consensus across a wide cross-section of the society. Drafts were discussed by various societal groups that included from Members of Parliament to farmers and ordinary people. The Vision document that now guides economic and social development efforts from the year 2000 to 2025, explicitly notes the important role that Information and Communications Technologies (ICTs) can play in economical development.

The necessary policy context is now in place to support the idea of a localized and freely available Operative System for Tanzania. Our mission is to create an Operating System which is easily understandable and freely available for common and normal Kiswahili speakers.

Free and Open Source Software

In brief, the Free and Open source software can be understood as the users' freedom to run, copy, distribute, study, change and improve the software.

The Free Software Foundation (FSF) refers in [3] to four kinds of freedom, for the users of the software:

- The freedom to run the program, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.

Up to now Free and Open Source Software remain foreign and inaccessible to majority of potential users in Tanzania and the Third World due to simple lack of competence in foreign languages used in the available software. Whereas many institutions fail to utilize the vast software resources available within the Free and Open Software community due to inability to adapt the available software to their local needs resulting from unavailable expertise. While information revolution proceeds many individuals are left aside on the Information Revolution due to financial constraints limiting their capacity to acquire lucrative commercial licenses to use closed source software.

Likewise many organizations capable of acquiring closed sources software off-shelf remain frustrated being unable to adapt their expensive software tools to ever changing business environments. Many developmental projects fear utilizing Free and Open Source software tools and platform due to lack of reliable support and maintenance partners. These phenomenons deprive these projects of the vast and precious resources available within Free and Open Source Software community. It is clear that advantages of adopting and sticking with Open Source are vast starting from free for all nature to opportunity of infinite customization and control of acquired software. The dynamic changing business environment calls for flexibility and configurable only available via Open Source software. The importance of Free and Open Source Software lies mainly on the inherent opportunity it offers of being adapted and consequently localized into any community.

In Tanzania, the available computing hardware capacity is highly underutilized. Most use the sophisticated computing machinery as if it were a mere electronic typewriter. This is a national epidemic as however much computing hardware is brought into the country, it stands to deliver no extras to ease people lives and improve living standards. Besides lack of proper training to the users, the main cause of this mishap is lack of specialized software for improving specific aspects of users productivity. Coupled with the narrow vision rendered by minimal commercial software that comes bundled with hardware, the utility of computing hardware is brought to a cul-de-sac ad infinitum.

Most consider the computing facilities as limited to simply what they found bundled with their hardware when they acquire it. Some, even if aware of some specialized software that they need to improve their productivity, more often the price and licensing costs required remain simply too much for them.

Still much more remain ignorant to the information machinery due to inability to comprehend the foreign languages used to access the facilities. Intimidated by the foreign tongue, many choose to stay away from the facilities and hence miss the possibilities brought forth through the information revolution. Free and Open Source software render itself as a remedy for above mentioned ills. Readily and freely available software offer chance to the average Tanzanian of utilization of computing facilities without guilty conscience of being pirate. The licenses coming with Free and Open Software explicitly allow use of the software without worrying of cash payments.

Toward a Localization Methodology

Localization (L10N) refers to the process of adapting a computer software and its cultural content to specific target audiences in specific locations. Software localization is not just the linguistic process of translation but a broader effort to take into consideration audiovisual, technical or legal requirements. Localization should be understood as a superordinate concept of translation.

Before starting any localization effort, it is necessary to assess which local resources are available and how to integrate them in a working methodology. In fact, there is a need to localize the localization methodology. Opposite to other localization efforts [4] where rich reference material is easily available including technical glossaries, spell-checkers, related computer literature or other localized softwares; that is not the case for Kiswahili. There is a need to design a localization methodology capable of both integrating the existing technical and linguistic resources and developing the missing ones.

The following section tries to describe a methodology for the localization of Open Source Software to Kiswahili. The methodology rather than just delivering some localized software (e.g. *OpenOffice*, *Mozilla*) aims to create a set of public online resources to encourage others to participate in the localization and development of software in Kiswahili.

Localization in 10 Phases

Select Software

The first phase is to decide what software(s) to localized (i.e. Where to start?). Different criteria needs to be considered here: target group, possible impact of the localized software, size of the translation, maturity and acceptance of the software, operative system requirements etc.

Three recommendations can be included here:

- Localization of cross-platform softwares as *OpenOffice* or *Mozilla* that run under Microsoft Windows and GNU Linux can be introduced faster.
- Localization of end-user applications and documentation should be the first priority. It is easier to assume that a person that can understand Kernel messages can speak better English than a person that wants just to write a document.
- Localization of end-user applications that do not require the use Internet should have priority.

A tentative Kiswahili Open Source Software localization roadmap could look like: Cross Platform Office Application (*OpenOffice*), Cross Platform Mail Client (*Mozilla*), Messengering Client (GAIM), Desktop (KDE). Including the software and the documentation.

Extract and convert strings

The first thing is to extract the strings that need to be translated out of the source code of the software. Different scenarios are possible depending on where and in which format we can find the strings.

- **Embedded in the source code.** This is the worse of the scenarios. The only way to localize to software is to edit the strings embedded in the source code and recompile the code.
- **(Un)known non-standard format.** In this scenario there is normally a specific tool to extract and face the translation. It is recommended to extract the strings and convert them to a known standard format that we can use in a common translation environment. The most common one is the *gettext* Portable Object Template (POT) format [5].
- **Documentation.** In this case, we need to rewrite the whole file trying to keep layout and structure. (i.e. Keeping the meta-code).
- **Portable Object Template.** Finally, if the software is using POT format for localization (e.g. KDE Desktop) we do not need to make any further conversions.

During this phase is recommended to extract a set of technical terms from the software to create a first basic glossary. The glossary should be translated as a joint effort between linguistics and computer scientists. It is also desirable to disseminate the glossary to get as much feedback as possible.

Translate strings

Before working in a group in a translation it is necessary to have a common accepted translation criteria. Consistency is one of the main problems when translation is done by many people. *Kbabel* under GNU/Linux [6] and *PoEdit* [7] (cross-platform Microsoft Windows/GNU Linux) are the most common tools to localize software that uses the POT format.

Kbabel supports the possibility of writing software plug-ins to assist translators. Here there is a list to some of the auxiliary resources that can be of great help if there are accessible from the translation environment.

- **Basic Glossary** A translation of the most common technical terms included in the software. The glossary needs to be available in advance or while the translation is done. It is important to have a mechanism to insert new terms into the glossary from the translation environment. Feedback can be retrieved by an online basic glossary and a mailing list for linguistics experts.
- **Extended Glossary** The extended glossary include not only the terms that appear in the software that is being localized but other related technical terms. Sources to build an extended glossary include: other softwares, other available glossaries, books and related technical literature. It might be of great interest to compile related technical literature in electronic format.
- **Spell-checker** *Kbabel* supports spell-checking via *Aspell* [8]. A spell-checker containing the new IT terms needs to be constantly updated and reachable from the translation environment.
- **Expert Advice** It is also necessary to describe a method for the translator to seek expert advice. A feedback channel between translators, computer science and linguistics experts needs to be available.
- **Other resources** Other auxiliary resources include: access to (online) dictionary, access to related technical literature, grammar checkers etc.

Review strings

In the first review phase, fuzzy translation or untranslated text will be fixed with external help if needed. In this phase it is necessary to examine consistency problems as acceleration keys or the need of localized audiovisual resources.

Insert strings

If the software does not support natively POT, the localized files needs to be converted to the original format and inserted back into the code.

Some extra changes (configuration) maybe needed in the software to be aware of a localized version. It is important to have in place a concurrent version system (CVS) that allows in any moment to identify the author and/or experts that worked in a given translation.

Create new localized code

In this phase we create the first release of the localized code. Normally the localized version can be either an add-on to the original code and/or a new binary version.

A bug report system should be available so feedback can be collected.

Review localized code

The second revision is done over the final working software. Having a revision of the working software allows to identify localization problems that can not be discovered by examining the pure translation of the strings.

Most of the problems are related to the context where the strings has been translated. The localized software should be distributed a group of users (testing group), internal and external

linguistic evaluation team and to technical evaluation team. All the feedback should be collected and changes require the go through phases 2 to 5 one more time.

The required documentation including user guides and help material should be translated by the end of this phase.

Create new localized code

Similarly to Phase 6 a final release of the localized code is built (add-on and/or binary version). The bug report system should be now available to any external users.

Publish localization

The Official release of the software should be distributed widely. Some of the activities that can be scheduled in this phase are: a public announcement including a press release/conference, newsletter to related mailing lists, workshop or kick-off training etc.

Maintenance of localized code

Every time that a new version of the software is released parts of the software needs to be re-localized again. Fortunately, part of existing material can be reuse to build the new version.

The maintenance will not be possible if an active group around the project has not been created. The group should also include volunteers and integrate the support from the governmental, educational and private sector.

Conclusions

Free and Open Source Software is available today for localization providing an excellent opportunity to suit potential local users. With a Kiswahili localized software many Tanzanians will no longer have to worry in learning an extra language for using the new technologies.

The localization roadmap and methodology outlined in this paper wants to serve Kiswahili speakers by introducing the open source working methodology into the ICTs.

Only a community that shares knowledge can shape the path to his own development.
Jamii pekee inayoweza kuchangia maarifa ndiyo inayoweza kutengeneza mazingira ya kujiletea maendeleo yake.

About the authors

Hashim Msafiri Twaakyondo is head of the Computer Science Department at the University of Dar es Salaam since 1999. Twaakyondo got his PhD in System Development at Shinshu University in Japan and is frequent reviewer and adviser in ICT policy and curriculum development. His research interests include programming languages and free and open source.

Alberto Escudero-Pascual got his PhD in Internet Security and Privacy at the Swedish Royal Institute of Technology (KTH). Escudero has been working since the early 90's as adviser and trainer in the areas of computer security, Internet infrastructure development and ICT policy.

References

- [1] **M.M. Mulokozi**, Kiswahili as a National Language, Institute for Asian and African Studies (IAAS), Finland. 2002.
- [2] **Tanzania Planning Commission**, The Tanzania National Development Vision 2025.
<http://www.tanzania.go.tz/vision.htm>
- [3] **GNU Project**, Definition of Free Software.
<http://www.gnu.org/philosophy/free-sw.html>
- [4] **Debian 110n Spanish**, Spanish Localization Effort. lists.debian.org/debian-110n-spanish/
- [5] **GNU Gettext Project**, Tools to produce multi-lingual messages.
<http://www.gnu.org/directory/localization/gettext.html>
- [6] **S. Visnovsky, M. Kiefer**, The Kbabel Handbook
<http://docs.kde.org/en/3.2/kdesdk/kbabel/>.
- [7] **V. Slavik**, Cross-platform gettext catalogs editor. <http://poedit.sourceforge.net>.
- [8] **K. Atkinson**, GNU Spell Checker <http://www.gnu.org/software/aspell/>.